Maxine Learning
Monsoon 2019
End-Semester Exam: CSE543/ECE5ML **20/11/2019**
Max. marks: 80 (+10 Extra credit) **Time Limit: 120 Minutes**

Instructions:
Please do not plagiarize. It will be dealt with very strictly!
Try to answer all questions. The last question is Extra Credit.
In the unlikely case that a question is ambiguous, please clearly state any assumptions that you are making. For reducing subjectivity in grading, please do this *even after clarifying* with the invigilator.
Good Luck!

---

1. (15 points) Decision Trees

   a) (5) We have seen decision trees for classification in class. How would you extend them to perform a regression task? What would be the advantage of doing so over linear regression?
   solution: We normally use entropy as a measure of impurity for classification with decision trees. For regression tasks, we can use Weighted Mean Square Error of the children nodes instead to model continuous variables.

$$MSE(t) = \frac{1}{N_t} \sum_{i \epsilon D_t} (y^{(i)} - \hat{y}_t)^2$$

   N_t is the number of samples at node t, D_t is the training subset at node t, y(i) is the true target value and $\hat{y}_t$ is the predicted target value. (3 marks, 1 mark for partial correct)

   Alternatively, modeling the probability of occurrence as a gaussian distribution, and calculate entropy based on this. Answers have been considered provided they are fully correct. Advantages over linear regression: (2 marks, any one point)

   - Decision trees support non linear decisions
   - Decision trees, being non parametric, support a large number of functional forms
   - No assumptions required about the underlying functions

   b) (1+2+2) What is entropy? Give the mathematical definition of entropy. Justify or refute the statement "A node should be a leaf node if its entropy is zero". Can you give one (or more) scenarios where you may choose to have a leaf node that has non-zero entropy?
   Entropy is defined as the expected number of bits needed to encode a randomly drawn value of a random variable X. It could also be defined as a measure of impurity, disorder or uncertainty in a set of samples. (0.5 marks + 0.5 for formulation)
   Entropy H(X) of a random variable X can be written as

$$H(X) = -\sum_{i=1}^{n} P(X = i) log_2 P(X = i)$$

   Justify: The entropy at a particular node will be 0 when the dataset S at that node is perfectly classified. There would be no point in splitting it further and would hence be a

leaf node.

Leaf node with non zero entropy: One case could be when the current dataset has data from multiple classes remaining, but there are no more attributes left to split the data on. Other cases could include degradation of performance due to splitting, pruning.

c) (5) Give a total of five advantages and disadvantages of decision trees over classifiers like SVMs or logistic regression. (1 mark for each point

Advantages -

- Decision trees are non parametric, and hence support a large number of functional forms
- Decision trees can be used for both continuous and categorical data
- Decision trees don't require much data processing (such as scaling, normalization, etc).
- Implicit feature selection

Disadvantages -

- Decision trees are much more prone to overfitting, prone to sampling errors
- Small changes in the data can cause a large change to the structure of the decision tree. In SVM or logisitic regression classifiers, changes to the data do no affect the decision boundary as extensively
- Exponential complexity in calculation which growing datasets.
- Tree splitting is locally greedy and can lead to an incorrect feature hierarchy which may lead to incorrect classifications

2. (20 points) Linear and Logistic Regression

a) (5) You wish to fit a polynomial to 2D data (x, y). You don't know what the right order of the polynomial might be. How would you go about fitting the model? Explain your overall strategy, as well as the maxine learning tool, the necessary loss function, transformations if any, and evaluation strategies.

b) (5) You have points along the surface of two concentric spheres in 3D, each belonging to a different class. How would you apply logistic regression to classify these samples?

c) (2+2+4+2) LASSO applies a certain kind of a regularization to introduce sparsity in the linear regression model parameters. Explain which regularization function is used, and what prior distribution does it assume over the model parameters? Derive the resulting loss function when the same prior distribution be used with logistic regression. Will it have a similar sparsity-inducing effect? Please explain your answer.

Solution:

a) Since the polynomial order is unknown, we will have to use cross-validation for model selection. Split the data into train, val and test sets and perform linear regression with increasing orders of transformations $\{x, x^2, x^3, \ldots\}$. Use validation error (MSE) to establish a good fit. Alternatively, an appropriate kernel could be used, but again, cross-validation should be used to establish the goodness of fit for the linear regression problem.

b) Use a transformation to map to radius (or alternatively use a second order transform, i.e., $\{x, y, z\} \Rightarrow \{x^2, y^2, z^2\}$, or a mapping to spherical coordinates). Since it is a sphere, cross terms are not necessary. A kernel would be an overkill as you know the structure of the data that you need to classify.

c) ℓ_1-norm is used for regularization, and it assumes a Laplacian or doubly exponential distribution on the parameters $p(\mathbf{w}) = \exp(-\lambda|\sum_i w_i|)$. Derivation of the loss is through the standard process of MAP estimation. The sparsity will be retained as the loss function is still convex and the priors on the parameters will act in the same manner as in LASSO.

3. (20 points) Support Vector Maxines and Kernel Methods

a) (2+2) The ξ^2 kernel is defined as $\kappa(x, y) = \sum_{i=1}^{D} \frac{x_i y_i}{x_i + y_i}$, where $x_i, y_i \geq 0$ and $\sum_{i=1}^{D} x_i = \sum_{i=1}^{D} y_i = 1$. Text data is often represented using feature vectors that are like histograms, frequently referred to as *bag of words*. Explain how would you use ξ^2 kernel to compute distances between two such feature points. Why is it justified to use this specific kernel? {*Hint*: Recall that kernel functions represent inner products between vectors in some feature space. How can you use inner products to compute distances?}

b) (2+6) What is complementary slackness with respect to a convex optimization problem? Does it apply to support vector maxines? Explain the implications of the complementary slackness condition by writing all the constraints on the primal and dual variables for the SVMs.

c) (4) Show that the constraints in a primal SVM formulation are affine, i.e., simultaneously convex and concave.

d) (4) How does the Lagrangian help in converting a constrained optimization problem to an unconstrained optimization problem. Starting from the original SVM formulation as a constrained optimization problem, develop the Lagrangian, and then the primal problem. Explain the equivalence of the primal problem with the original SVM formulation with constraints.

Solution:

a) (2 points) - Compute distane between 2 features.

$K(x, y) = \sum_{i=1}^{D} \frac{x_i y_i}{x_i + y_i}$
where $x_i, y_i \geq 0$ and $\sum_{i=1} Dx_i = \sum_{i=1}^{D} y_i = 1$

Kernel function represent dot products between vectors in some feature space.
Eucledian distance between 2 points can be calculated as:

$$D(x, y) = \sum_{i=1}^{D} \|X_i - Y_i\|^2$$

$$D(x, y) = \sum (X_i - Y_i)(X_i - Y_i)^T$$

$$D(x, y) = \sum X_i^2 + Y_i^2 - 2X_i Y_i$$

$$D(x, y) = \sum X_i^T X_i + Y_i^T Y_i - 2X_i^T Y_i$$

$$D(x,y) = \sum_{i=1}^{P} K(X_i, X_i) + K(Y_i, Y_i) - 2K(X_i, Y_i)$$

(2 points) - Justification.
It is justified to use this specific kernel because $K(x,y) = \sum_{i=1}^{D} \frac{x_i y_i}{x_i + y_i}$ when X_i, Y_i are close, the value of $K(x_i, y_i)$ reduces and when the vectors are far away, the value of $K(x_i, y_i)$ increases.
The kernel matrix can be seen as the measure of similarity between 2 given vectors of the above kernel aligns with that intuition since the kernel values increase where x and y are similar and decrease when they are different.

b) Suppose (α,β) are dual feasible. $\theta_D(\alpha, \beta) = nf(x^*) + \sum_{i=1}^{n} \alpha_i g(x_i^*) + \sum_{i=1}^{P} \beta_i h(x_i^*)$

(2 points)
Complementary slackness states that if strong duality holds, then $\alpha_i g(x_i^n) = 0$ for each $i = 1, 2, ...m$

It can be written as phase pair of conditions:

$$\alpha_i^* > 0 \Rightarrow g_i(x^*) = 0$$
$$g_i(x^*) < 0 \Rightarrow \alpha_i^* = 0$$

(1 + 1 point)
Whenever α_i is strictly greater than zero, then this implies that the corresponding inequality constraint must hold with equality. We refer this an active constraint. In case of SVM, active constraint are support vectors. Thus, **this apply to SVM.**

KKT Complementary requires that for any primal optimal (w^*, b^*, ζ^*) and dual optimal (α, β)

$$\alpha_i^*(1 - \zeta^* - y^{(i)}(w^{*T} x^{(i)} + b^*)) = 0$$
$$\beta^* \zeta^* = 0$$

(4 points)
For $i = 1....m$, from the first condition we can see that if $\alpha_i^* > 0$ then in order for the product to be zero then $1 - \zeta^* - y^{(i)}(w^{*T} x^{(i)} + b^*) = 0$

It follows that $y^{(i)}(w^{*T} x^{(i)} + b^*) \leq 1$ Since $\zeta^* \geq$ by primal feasibility. Similarly of $\beta_i^* > 0$, then $\zeta^* = 0$ to ensure complementary. From the primal constraint $y^{(i)}(w^{*T} x^{(i)} + b^*) \geq 1 - \zeta_i$ it follows that $y^{(i)}(w^{*T} x^{(i)} + b^*) \geq 1$ Finally since β equivalent to $\alpha_i < C$ since $\alpha_i + \beta_i = C$ we can summarize KKT conditions. $\alpha_i^* < C \Rightarrow y^{(i)}(w^{*T} x^{(i)} + b^*) \geq 1$ $\alpha_i^* > 0 \Rightarrow y^{(i)}(w^{*T} x^{(i)} + b^*) \leq 1$

or equivalently

$$\alpha_i^* = 0 \Rightarrow y^i(W^T x^i + b^*) \geq 1$$
$$0 \leq \alpha_i^* \leq C \Rightarrow y^i(W^T x^i + b^*) = 1$$
$$\alpha_i^* = C \Rightarrow y^i(W^T x^i + b^*) \leq 1$$

Complementary slackness conditions:

$$\alpha_j^*[y_j(\vec{w}^*.\vec{x}_j + b) - 1 + \zeta_j] = 0 \Rightarrow \alpha_j^* = 0 \vee y_j(\vec{w}^*.\vec{x}_j + b) = 1 - \zeta_j$$

c) (3 points)
Primal formulation:

$$1 - \zeta^* - y^{(i)}(w^{*T}x^{(i)} + b^*) \leq 0$$
$$-\zeta \leq 0$$
$$c_1 = 1 - \zeta^* - y^{(i)}(w^{*T}x^{(i)} + b^*)$$
$$c_2 = -\zeta$$

Double Differentiate:

$$\frac{\partial c_1}{\partial w_j} = -y^i x_j^i$$
$$\frac{\partial^2 c_1}{\partial w_j^2} = 0$$

Hessian matrix would be constant positive matrix.

$$\frac{\partial^2 c_2}{\partial w_j^2} = constant$$

(1 point)
Therefore constraints are both convex and concave thus affine.

d) (3 points)

$$minimize\ \frac{1}{2}||w||^2$$
$$Constraint:\ y_i(w^T x_i + b) \geq 1$$

Primal formulation is as follows:

$$L_p(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i(y_i(w^T x_i + b) - 1)$$

$$s.t.\ \alpha_i \geq 0$$

Taking derivatives we get:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i\ and\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

Applying these values in the original primal formulation we get the dual formulation.

$$max\ \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$Conditioned\ on\ \alpha_i \geq 0\ and\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

(1 point)
Equivalence from primal to dual on the dot product (because we used the lagrangian).

4. (25 points) Neural Networks

a) (5) Give the expression for binary cross-entropy (BCE) loss. Usually, it is used with a sigmoid output activation. Argue why is the BCE loss more effective than the mean-squared error (MSE) loss. Make use of corresponding gradients to support your arguments.
Solution:
Expression for BCE

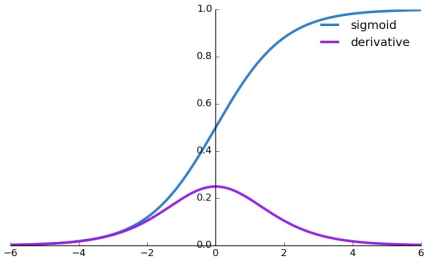
$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p(y_i)) + (1-y_i)\log(1-p(y_i)) \tag{1}$$

(1 marks)
If we use BCE instead of MSE loss, learning can occur much faster during the beginning phases of training.
Proof: Consider we have a binary classification setup, where input feature is x_i, ground truth label is y_i and $\hat{y}_i = h_\theta(x_i) = \sigma(Wx_i + b)$ is the corresponding prediction.

$$\text{MSE LOSS} : J = \frac{1}{2}(y_i - \hat{y}_i)^2 \tag{2}$$

$$\frac{\partial J}{\partial W} = (y_i - \hat{y}_i)\sigma(Wx_i + b)(1 - \sigma(Wx_i + b)) \tag{3}$$

(1 marks)
Since we initialized our weights randomly with values close to 0, this expression in equation (3) will be very close to 0, which will make the partial derivative nearly vanish during the early stages of training. As you can see in the figure **??** gradients are small whenever outputs are close to 0 or 1. This can lead to slower learning at the beginning stages of gradient descent, since the smaller derivatives change each weight by only a small amount, and gradient descent takes a while to get out of this loop and make larger updates towards a minima.
(1 marks)
On the other hand, if we use BCE loss:



$$\text{BCE LOSS} : -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\sigma(Wx_i + b)) + (1-y_i)\log(1-\sigma(Wx_i + b)) \tag{4}$$

$$\frac{\partial J}{\partial W} = x_i(\sigma(Wx_i + b) - y_i) \tag{5}$$

(1 marks)
We can see from equation (5) the magnitude of gradient completely depends on the magnitude of the error $(\sigma(Wx_i + b) - y_i)$–i.e how far our prediction is from the ground truth. It means that during early phase of learning, the error magnitude would be larger, hence fast learning, while later on magnitude would be smaller which leads of smaller derivatives , and hence corresponding to smaller adjustments to the weight variables.
(1 marks)

b) (3+1+1) The `tanh` activation is given by $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$. Compute the derivative of this activation function, and explain how does it compares to that of sigmoid? What could be the possible advantages of the tanh activation over sigmoid? Can a tanh activation be used at the output layer of a binary classifier? Explain why or why not.
Solution:

$$\text{Derivative of tanh:} \qquad \tanh' = 1 - \tanh(x)^2 \tag{6}$$

- Like sigmoid, tanh is also a sigmoidal -(S-shaped) function.
- The derivative of tanh includes the tanh function itself, the caching trick (same as sigmoid) can be used for layers that implement tanh activation functions.
- The derivatives are steeper for tanh than sigmoid.
- Sigmoid maps strong-negative inputs to values close to zero, because of this neural network get stuck during training. While, tanh maps strongly negative inputs to the negative outputs and only zero-valued inputs are mapped to near-zero outputs. These properties make the network less likely to get stuck during training.

Yes, we can use tanh activation at the output layer of binary classification.

c) (5) How would you extend the binary cross-entropy loss to multiple classes? What kind of activation would you have in the output layer of a neural network designed for multi-class classification?
Solution: The logistic function can be used for the classification between two target classes $t = 1$ and $t = 0$, this can be generalized to output a multiclass categorical probability distribution by the softmax function.

(1 marks)
The softmax function S takes $C-$dimensional vector x as input and outputs a $C-$dimensional vector y of real values between 0 and 1. Mathematically softmax function is defined as below:

$$y_c = S(\mathbf{x})_c = \frac{e^{x_c}}{\sum_{d=1}^{C} e^{z_d}} \quad \text{for} \quad c \in \{1...C\} \tag{7}$$

$$\begin{bmatrix} p(t=1|\mathbf{x}) \\ . \\ . \\ p(t=C|\mathbf{x}) \end{bmatrix} = \begin{bmatrix} S(\mathbf{x})_1 \\ . \\ . \\ S(\mathbf{x})_C \end{bmatrix} = \frac{1}{\sum_{d=1}^{C} e^{z_d}} \begin{bmatrix} e^{z_1} \\ . \\ . \\ e^{z_C} \end{bmatrix} \tag{8}$$

Where $P(t = x|\mathbf{x})$ is the probability that the class is c given the input x.
(2 marks)
Cross-entropy loss for softmax function:

The likelihood $\ell(\theta|\mathbf{t},\mathbf{x})$ can be written as a conditional distribution $P(\mathbf{t},\mathbf{x}|\theta)$ can be written as:

$$P(\mathbf{t}|\mathbf{x},\theta) = P(\mathbf{t}|\mathbf{x},\theta)P(\mathbf{x}|\theta) \tag{9}$$

For a fix θ and since each t_i is dependent on the $\mathbf{x}$ and only 1 class can be activated in the $\mathbf{t}$ we can write.

$$P(\mathbf{t}|\mathbf{z}) = \prod_{i=c}^{C} P(t_c|\mathbf{x})^{t_c} = \prod_{i=c}^{C} S(\mathbf{x})^{t_c} = \prod_{i=c}^{C} (y_c)^{t_c} \tag{10}$$

$$-\log \ell(\theta|\mathbf{t},\mathbf{x}) = \xi(\mathbf{t},\mathbf{x}) = -\log \prod_{i=c}^{C} (y_c)^{t_c} = \sum_{i=c}^{C} t_c \log(y_c). \tag{11}$$

Which is the cross entropy function ξ. You may verify that for two class system output $t_2 = 1 - t_1$. The error function become $\xi(\mathbf{t},\mathbf{y}) = -t_c \log(y_c) - (1 - t_c)\log(1 - y_c)$. The cross-entropy error function over a batch of multiple samples of size n can be calculated as:

$$\xi(T,Y) = \sum(\mathbf{t}_i,\mathbf{y}_i)_{i=1}^{n} = -\sum_{i=1}^{n}\sum_{i=c}^{C} t_{ic}\log(y_{ic}) \tag{12}$$

Where t_{ic} is 1 if and only if sample i belongs to class c and y_{ic} is the output probability that sample i belongs to class c.
(2 marks)

d) (5) The first layer of AlexNet has 96 filters of size 11×11, that process an input image of $224 \times 224 \times 3$ to generate feature maps of size 55×55. What is the stride used in the first layer? The last max pooling layer of size $(13 \times 13 \times 256)$ is connected to a two layered fully-connected layer of size 4096, followed by a 1000-node softmax based output layer. What is the total number of parameters for the last four layers (max pooling, linear, linear and softmax)?
Solution:
Stride : 3.94 Assuming $224 \times 224 \times 3$ / 4 assuming $227 \times 227 \times 3$
(1 marks) No. of parameters (max pool) : 0
(1 marks) No. of parameters (FC1) : 37,752,832
(1 marks) No. of parameters (FC2) : 16,781,312
(1 marks) No. of parameters (softmax) : 4,097,000

f) (1+2+2) What is the rectified linear unit activation? Give it's mathematical form and explain why is it expected to work better in deeper networks than sigmoid or tanh? What is dropout, and why is it effective in reducing overfitting?
Solution:
The Rectified Linear Unit is the most commonly used activation function. The function returns 0 if it receives any negative input, for any positive value x it returns that value back.
(1 marks)
The main problem with Sigmoid and tanh is that they saturate. During backpropagation, they may produce a gradient of zero for large inputs, which can slow or even halt the

learning process. The ReLU function $f(x) = max(0, x)$ doesn't saturate for large positive inputs.
The ReLU function has a derivative of 0 over half it's range (the negative inputs). For positive inputs, the derivative is 1. When training on a reasonable sized batch, there will usually be some data points giving positive values to any given node. So the average derivative is rarely close to 0, which allows gradient descent to keep progressing.
(2 marks)
Dropout is a technique for addressing the overfitting problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. (2 marks)
For detail please refer **this** docuement.

5. (10 points) Overfitting and Underfitting

   a) (4) Adding a regularizer to a linear regression model results in reducing the variance and increasing the bias. Do you agree with this statement? Justify your answer.
   Yes, adding a regularizer will help. (1 mark)
   This helps in lowering down the model complexity by imposing sparsity in the weights. Consider for example, imposing LASSO and Ridge regression leads to reducing most of the weights / coefficients to zero or reducing their magnitude (closer to zero) respectively. This helps in reducing the variance by making the estimator simpler. (2 marks)
   There is an increase in the bias due to the additional information imposed. (1 mark)

   b) (2) What strategies would you use to avoid overfitting in deep neural networks?
   Write any two points: (1 mark for each point)

   - Adding dropout layers.
   - Feature selection
   - Residual connections
   - Data augmentation
   - Regularization / Early stopping

   c) (4) How would you determine if your model is underfitting?

   - Check if the training error is quite high.
   - Check if the difference between the training and test errors is low. (2 marks)
   - Does the performance remain the same upon increasing the data size?
   - Does changing / increasing the feature set help in refining the performance?